



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

Address: COMMISSIONER FOR PATENTS

P.O. Box 1450

Alexandria, Virginia 22313-1450

www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/521,586	09/26/2005	Joachim Roos	1554-1002	4031
<div>466                      7590                      04/24/2009</div> <div>YOUNG &amp; THOMPSON</div> <div>209 Madison Street</div> <div>Suite 500</div> <div>ALEXANDRIA, VA 22314</div>				
EXAMINER				
VICARY, KEITH E				
ART UNIT		PAPER NUMBER		
2183				
MAIL DATE		DELIVERY MODE		
04/24/2009		PAPER		

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

# Office Action Summary

**Application No.**

10/521,586

**Applicant(s)**

ROOS ET AL.

**Examiner**

Keith Vicary

**Art Unit**

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --  
**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 19 March 2009.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1, 3, 5-8, 10 and 12-20 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1, 3, 5-8, 10, and 12-20 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO/SF-08)  
Paper No(s)/Mail Date \_\_\_\_\_
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: \_\_\_\_\_

## **DETAILED ACTION**

### ***Continued Examination Under 37 CFR 1.114***

0. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 3/19/2009 has been entered.

1. Claims 1, 3, 5-8, 10, and 12-20 are pending in this application and presented for examination. Claims 1, 8, and 17 are newly amended, and claims 2, 4, 9, and 11 are newly cancelled by amendment filed 3/19/2009.

### ***Claim Objections***

2. Claims 1, 8, and 17 are objected to because of the following informalities.

Appropriate correction is required.

- a. In claim 1, lines 31-32, "wherein the execution of the microcode program being dependent upon the first request code in that the start address..." should presumably be something akin to "wherein the execution of the microcode program is dependent upon the first request code in that the start address..."  
Also see claims 8 and 17.

***Claim Rejections - 35 USC § 112***

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 3, 10, and 17-20 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

5. Claims 3 and 10 recite the limitation "[a] processor according to claim 2" and "[a] method according to claim 9" respectively. However, claims 2 and 9 are newly cancelled; therefore, it is indefinite as to which claim the aforementioned claims are dependent on.

6. Claim 17 recites the limitation "the access points providing access to the pipeline to at least one external device" in lines 5-7. It is indefinite as to what is being conveyed, examiner recommends reciting of, for example, access from the pipeline to at least one external device.

- b. Claims 18-20 are rejected for failing to alleviate the rejection of claim 17 above.

***Claim Rejections - 35 USC § 103***

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and

the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

8. Claims 1, 3, 5-8, 10, 12-16, and 19-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cataldo (Net processor startup takes pipelined path to 40 Gbits/s) in view of Dorst (20040098549 A1) in view of FOLDOC (definition of microcode) in view of Tanenbaum (Structured Computer Organization) in view of Harriman (US 6330645 B1).

9. Consider claims 1 and 8, Cataldo discloses a programmable pipeline adapted to directionally transfer data packets through the pipeline from a first end of the pipeline to a second end of the pipeline, and adapted to perform sequences of instructions on the data packets, the pipeline comprising plural pipeline stages being arranged in a row between the first end of the pipeline and the second end of the pipeline (Page 1, second to fifth paragraphs, programmable pipeline architecture...PISC can be programmed to do tasks...each PISC stage acts as a mini-processor).

However, Cataldo does not disclose the pipeline comprises plural access points being arranged in a row between the first end of the pipeline and the second end of the pipeline, the access points providing at least one external device with access to the pipeline, at least one of the access points separating and connecting two of the pipeline stages, and does not disclose at least one interface engine comprising a microcode memory and being adapted to be connected to at least one external device located externally of the processor; the at least one interface engine is connected to the plural access points, and wherein the at least one interface engine is adapted to receive a request from any one of the connected access points of the programmable pipeline, the

request being received upon arrival of one of the data packets at the respective any one access point, the request comprising a first request code, according to a first coding scheme adapted for the processor, execute a microcode program stored in the microcode memory, the execution of the microcode program being dependent upon the first request code, to obtain, as a result of the execution of the microcode program, at least one device control code according to a second coding scheme adapted for the external device, wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code, generate at least one request output, the at least one request output being based at least partly on the request from the one access point and/or at least partly on the device control code, send at least one request output to the external device, receive, responsive to the request output, an external reply from the external device, and send to the pipeline a response, based on the external reply, to the request. Cataldo also does not disclose that the interface engine includes an arbiter configured to allow forwarding of requests from the connected access points in a fair manner between each of the connected access points.

On the other hand, Dorst does disclose at least one interface engine (Figure 3, memory controller 1005, which includes interface circuitry 4010 as shown in Figure 4) being adapted to be connected to at least one external device located externally of the processor (Figure 3 shows multiple memories 1015 external to the processor and connected to the data processing block 3005, which includes the memory controller); wherein the at least one interface engine is connected to plural access points, ([0031],

several processors may share a memory controller; each processor is an access point or contains an access point), and wherein the at least one interface engine is adapted to receive a request from any one of the connected access points ([0010], memory controller couples to the processors and to the memories, and provides communication between the processor and memories; it is inherent that this communication includes a request for data in a memory by the processor), the request comprising a first request code, according to a first coding scheme adapted for the processor (it is inherent that a data instruction such as a load or store has parameter bits to indicate the address or memory module that needs to be accessed; these bits correlate to the request code and its format to be recognized is the coding scheme; note that this scheme is adapted for the processor in that it does not contain any memory-specific parameters, which are located in the memory controller), and, dependent upon the first request code, obtaining at least one device control code according to a second coding scheme adapted for the external device (third sentence of the abstract, the interface circuitry communicates with the memory by providing a plurality of control signals; as explained above, certain control signals, correlating to device control codes, are asserted at certain times. This second coding scheme is the code that is outputted to the memory which controls the memory, as opposed to the first coding scheme which is the code that makes up the instruction and does not include these memory access signals such as the read enable signal. These signals are depending upon the first request code as follows. [0042] discloses that each memory may correspond to a respective register set, and [0039] discloses each register set stores read timing-parameters and write timing-parameters).

Therefore, the device control codes are obtained depending upon what memory is to be read or written. Given that it is inherent that a memory-access instruction from a processor cannot randomly pick a memory to be read or written to and still maintain correct program execution, the memory access instruction, which is or contains the first request code, must in some way specify which memory is to be accessed and thus the corresponding device control codes), generate at least one request output, the at least one request output being based at least partly on the request from the one access point and/or at least partly on the device control code (Figure 6, control signals, explained further in [0045] and [0046]; it is inherent that which of the external devices gets the control signals is based upon the data address or other parameter in the memory access instruction, and to which memory the memory controller sends the read/write enable signal, for example), send at least one request output to the external device (Figure 6, control signals, explained further in [0045] and [0046]), receive, responsive to the request output, an external reply from the external device ([0040], memory retrieves the data and makes it available through data bus 4020; the newly retrieved data is the external reply), and send to the access point a response, based on the external reply, to the request (Figure 1, given the memory controller connects the processor and memories together, it is inherent that the retrieved data would be passed on to the pipeline).

The teaching of Dorst reduces the burden and overhead of processors interfacing with and controlling the memory (Dorst, [0003]), as well as flexibly controls a multitude of memory circuits in a simple-to-use manner (Dorst, [0006]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Dorst with the invention of Cataldo in order to reduce the burden and overhead of processors interfacing with and controlling the memory, as well as flexibly control a multitude of memory circuits in a simple-to-use manner. Note that the teaching of Dorst, when applied to the invention of Cataldo, teach the overall limitation that the pipeline comprises plural access points being arranged in a row between the first end of the pipeline and the second end of the pipeline, the access points providing at least one external device with access to the pipeline, at least one of the access points separating and connecting two of the pipeline stages, wherein the request is received from any one of the connected access points of the programmable pipeline, the request being received upon arrival of one of the data packets at the respective any one access point (a PISC stage of Cataldo which connects to the memory controller of Dorst would be the access point, and this PISC stage would be arranged between the first end of the pipeline and the second end of the pipeline, provide data from a memory to the pipeline, and itself be between two other PISC stages).

Dorst discloses using hardware in the interface engine to obtain device control codes from the first request code ([0070] and [0071] discloses using finite state machines, counters, and programmable registers in order to implement the control circuitry for the memory controller and establish relative timing relationships among control signals and address signals). Consequently, Cataldo and Dorst do not disclose *of executing a microcode program stored in a microcode memory* which obtains device

control codes from the first request code, *wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code.*

On the other hand, FOLDOC discloses of microcode (definition of microcode).

Tanenbaum discloses that hardware and software are logically equivalent, and decisions to put certain functions in hardware and other in software are based on such factors as cost, speed, reliability, and frequency of expected changes (pages 11-12 in whole).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of FOLDOC with the invention of Cataldo and Dorst using the motivation of Tanenbaum in order to, for example, save on cost and support changes in the operation of the processor. Note that the teaching of FOLDOC, when applied to the invention of Cataldo and Dorst, teach the overall limitation of executing a microcode program stored in a microcode memory, wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code, and is explained as follows. As explained above, the hardware of Dorst outputs device control codes dependent upon the memory which is to be read or written (which is indicated in some form by the request codes). When the hardware of Dorst is replaced by the microcode of FOLDOC, the combination would entail microcode outputting control codes dependent upon these request codes. Therefore, the execution of the microcode (which necessarily involves executing a first instruction of the microcode,

which is at a start address) is dependent upon receiving a request code. Moreover, the execution of a specific microcode (which must specifically reference the particular register set with timing parameters that correlates to the memory to be accessed) is dependent upon a particular request code (which denotes the memory to be accessed). Finally, it is inherent that microcode is stored in some form of microcode memory in order to be run. Therefore, the combination teaches the overall limitations of executing a microcode program stored in a microcode memory, wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code.

Dorst discloses in [0031] that several processors may share a memory controller as persons of ordinary skill in the art would understand; however, Cataldo, Dorst, FOLDDOC, and Tanenbaum do not explicitly disclose that the interface engine includes an arbiter configured to allow forwarding of requests from the plural access points in a fair manner between each of the plural access points.

Although the examiner believes it would have been readily recognized to one of ordinary skill in the art at the time of the invention that there must be some form of arbitration given that multiple processors are sharing a memory controller which cannot handle all their requests at one time, Harriman nevertheless discloses of an arbiter configured to allow forwarding of requests from plural access points in a fair manner between each of the plural access points (col. 4, lines 19-40, arbiter 224, round robin scheme).

Harriman's teaching allows for forwarding of requests from plural access points to be done in a fair manner (Harriman, as cited above, col. 4, lines 19-40) to provide equal access to different requesters (Harriman, col. 4, lines 29-32).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Harriman with the invention of Cataldo, Dorst, FOLDLOC, and Tanenbaum in order to provide equal access to different requesters.

10. Consider claims 3 and 10, Dorst discloses the device control code is an operational code of the external device ([0046], the control signals serve to operate the external device).

11. Consider claims 5 and 12, Dorst discloses the interface engine comprising a reply control unit (figure 3, memory controller 1005 still) adapted to receive at least one receiver ID signal related to the request, and to determine, based on the receiver ID signal, the access point which is to receive the response (it is inherent that given a plurality of processors sharing the same memory controller, a processor which requests data from memory for itself would be the one to actually receive data. Because of this, it is further inherent that there must be an ID signal, either explicitly given by the processor identifying itself or implicit in that merely sending a memory instruction to the memory controller is an implicit signal as the memory controller is capable of identifying

which processor sent that memory instruction, perhaps due to the pins at which it was received on).

12. Consider claims 6 and 13, Dorst discloses the reply control unit is adapted to receive an input control signal ([0010], memory controller couples to the processor and to the memories, and provides communication between the processor and memories; it is inherent that this communication includes a request for data in a memory by the processor; this request for data correlates to the input control signal, as the request for data is inputted into the memory controller in order to control the memory into outputting the data), and timing information for receiving the external reply from the external device is determined from the input control signal ([0071] discloses that programmable registers provide a mechanism for timing among control signals, and [0039] specifically discloses that the programmable registers store read timing-parameters and write timing-parameters. Because [0042] discloses that each memory may correspond to a respective register set, it is inherent that, once a memory instruction occurs and its relevant parameters are sent to the memory controller, the bits regarding the specific address or memory to be accessed are used in some form to select the register with the timing-parameters relevant to that specific address or memory. These timing-parameters are used by the finite state machine to execute the program, as certain control signals are asserted at certain times to certain memory because of the parameters/code of the memory instruction. One of these control signals, as seen in

Table 1 near [0053], is read-enable pulse-width, which is used to program the number of cycles the read-enable signal remains asserted during read operations).

13. Consider claims 7 and 14, Dorst discloses the number of access points adapted to send a request to the interface engine can be adjusted ([0031], which first discloses that a system may have more than one processor and more than one memory controller. It then discloses that several processors *may* share a memory controller. If several processors *may* share a memory controller, then it is possible that several processors *do not* share a memory controller; and if a given processor is not able to access all memory controllers, then it is apparent that they cannot send a request to that memory controller either. Furthermore, because it is disclosed that several processors *may* share a memory controller, or vice-versa, it is inherent that which is actually the case is adjustable).

14. Consider claims 15, 16, and 20, Harriman discloses the fair manner of the arbiter is a round-robin manner between the plurality of access points (col. 4, lines 19-40, arbiter 224, round robin scheme).

15. Consider claim 19, although Dorst discloses in [0031] that several processors may share a memory controller as persons of ordinary skill in the art would understand, Cataldo, Dorst, FOLDOC, and Tanenbaum do not explicitly disclose that the interface

engine includes an arbiter configured to allow forwarding of requests from the plurality of access points in a fair manner between each of the plurality of access points.

Although the examiner believes it would have been readily recognized to one of ordinary skill in the art at the time of the invention that there must be some form of arbitration given that multiple processors are sharing a memory controller which cannot handle all their requests at one time, Harriman nevertheless discloses of an arbiter configured to allow forwarding of requests from plural access points in a fair manner between each of the plural access points (col. 4, lines 19-40, arbiter 224, round robin scheme).

Harriman's teaching allows for forwarding of requests from plural access points to be done in a fair manner (Harriman, as cited above, col. 4, lines 19-40) to provide equal access to different requesters (Harriman, col. 4, lines 29-32).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Harriman with the invention of Cataldo, Dorst, FOLDOC, and Tanenbaum in order to provide equal access to different requesters.

16. Claim 17 is rejected under 35 U.S.C. 103(a) as being unpatentable over Cataldo, Dorst, FOLDOC, and Tanenbaum.

17. Consider claim 17, Cataldo discloses a programmable pipeline comprised of plural pipeline stages located between a first end of the pipeline and a second end of

the pipeline (Page 1, second to fifth paragraphs, programmable pipeline architecture...PISC can be programmed to do tasks...each PISC stage acts as a mini-processor; these access points are each processor or its corresponding ports), the pipeline configured to directionally transfer data packets from the first end to the second end (Page 1, second to fifth paragraphs, programmable pipeline architecture...PISC can be programmed to do tasks...each PISC stage acts as a mini-processor).

However, Cataldo does not disclose that the pipeline comprises access points being located between a first end of the pipeline and a second end of the pipeline, the access points providing access to the pipeline to at least one external device, at least some of the access points separating and connecting an adjacent two of the pipeline stages, and does not disclose of an interface engine, comprising a microcode memory, having a reply control unit, and connected to an external device located externally of the processor, and connected to the plurality of access points, the interface engine configured to receive a request from the pipeline upon arrival of a data packet at any one of the connected access points, the request comprising a first request code, according to a first coding scheme adapted for the processor, execute a microcode program stored in the microcode memory, the execution of the microcode program being dependent upon the first request code, to obtain, as a result of the execution of the microcode program, at least one device control code according to a second coding scheme adapted for the external device, wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code, generate at

least one request output, the at least one request output being based at least partly on the request from the one access point and/or at least partly on the device control code, send the at least one request output to the external device, receive, responsive to the at least one request output, an external reply from the external device, and send a response, based on the external reply, to the pipeline, and the reply control unit configured to i) receive a receiver ID signal related to the request, and ii) determine, based on the receiver ID signal, the any one of the plurality of access points of the pipeline to receive the response.

On the other hand, Dorst does disclose an interface engine (figure 3, memory controller 1005, which includes interface circuitry 4010 as shown in figure 4), having a replay control unit (figure 3, memory controller 1005 still), and connected to at least one external device located externally of the processor (figure 3 shows multiple memories 1015 external to the processor and connected to the data processing block 3005, which includes the memory controller), and connected to the plurality of access points, ([0031], several processors may share a memory controller; each processor is an access point or contains an access point), the interface engine configured to i) receive a request from one of the connected access points ([0010], memory controller couples to the processors and to the memories, and provides communication between the processor and memories; it is inherent that this communication includes a request for data in a memory by the processor), the request comprising a first request code, according to a first coding scheme adapted for the processor (it is inherent that a data instruction such as a load or store has parameter bits to indicate the address or memory module that

needs to be accessed; these bits correlate to the request code and its format to be recognized is the coding scheme; note that this scheme is adapted for the processor in that it does not contain any memory-specific parameters, which are located in the memory controller), and, dependent upon the first request code, obtaining at least one device control code according to a second coding scheme adapted for the external device (third sentence of the abstract, the interface circuitry communicates with the memory by providing a plurality of control signals; as explained above, certain control signals, correlating to device control codes, are asserted at certain times. This second coding scheme is the code that is outputted to the memory which controls the memory, as opposed to the first coding scheme which is the code that makes up the instruction and does not include these memory access signals such as the read enable signal. These signals are depending upon the first request code as follows. [0042] discloses that each memory may correspond to a respective register set, and [0039] discloses each register set stores read timing-parameters and write timing-parameters. Therefore, the device control codes are obtained depending upon what memory is to be read or written. Given that it is inherent the a memory-access instruction from a processor cannot randomly pick a memory to be read or written to and still maintain correct program execution, the memory access instruction, which is or contains the first request code, must in some way specify which memory is to be accessed and thus the corresponding device control codes), generate at least one request output, the at least one request output being based at least partly on the request from the one access point and/or at least partly on the device control code (Figure 6, control signals, explained

further in [0045] and [0046]; it is inherent that which of the external devices gets the control signals is based upon the data address or other parameter in the memory access instruction, and to which memory the memory controller sends the read/write enable signal, for example), send the at least one request output to the external device (Figure 6, control signals, explained further in [0045] and [0046]), receive, responsive to the at least one request output, an external reply from the external device ([0040], memory retrieves the data and makes it available through data bus 4020; the newly retrieved data is the external reply), and send to the access point a response, based on the external reply, to the request (Figure 1, given the memory controller connects the processor and memories together, it is inherent that the retrieved data would be passed on to the pipeline), and the reply control unit configured to i) receive a receiver ID signal related to the request, and ii) determine, based on the receiver ID signal, the any one of the plurality of access points of the pipeline to receive the response (it is inherent that given a plurality of processors sharing the same memory controller, a processor which requests data from memory for itself would be the one to actually receive data. Because of this, it is further inherent that there must be an ID signal, either explicitly given by the processor identifying itself or implicit in that merely sending a memory instruction to the memory controller is an implicit signal as the memory controller is capable of identifying which processor sent that memory instruction, perhaps due to the pins at which it was received on).

The teaching of Dorst reduces the burden and overhead of processors interfacing with and controlling the memory (Dorst, [0003]), as well as flexibly controls a multitude of memory circuits in a simple-to-use manner (Dorst, [0006]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Dorst with the invention of Cataldo in order to reduce the burden and overhead of processors interfacing with and controlling the memory, as well as flexibly control a multitude of memory circuits in a simple-to-use manner. Note that the teaching of Dorst, when applied to the invention of Cataldo, teach the overall limitation that the pipeline comprises access points located between a first end of the pipeline and a second end of the pipeline, the access points providing access to the pipeline to at least one external device, at least one of the access points separating and connecting an adjacent two of the pipeline stages, the request being received upon arrival of one of the data packets at any one of the connected access points, and sending a response, based on the external reply, to the pipeline (a PISC stage of Cataldo which connects to the memory controller of Dorst would be the access point, and this PISC stage would be arranged between the first end of the pipeline and the second end of the pipeline, provide data from a memory to the pipeline, and itself be between two other PISC stages).

Dorst discloses using hardware in the interface engine to obtain device control codes from the first request code ([0070] and [0071] discloses using finite state machines, counters, and programmable registers in order to implement the control circuitry for the memory controller and establish relative timing relationships among

control signals and address signals). Consequently, Cataldo and Dorst do not disclose of *executing a microcode program stored in a microcode memory* which obtains device control codes from the first request code, *wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code.*

On the other hand, FOLDOC discloses of microcode (definition of microcode).

Tanenbaum discloses that hardware and software are logically equivalent, and decisions to put certain functions in hardware and other in software are based on such factors as cost, speed, reliability, and frequency of expected changes.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of FOLDOC with the invention of Cataldo and Dorst using the motivation of Tanenbaum in order to, for example, save on cost and support changes in the operation of the processor. Note that the teaching of FOLDOC, when applied to the invention of Cataldo and Dorst, teach the overall limitation of executing a microcode program stored in a microcode memory, wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code, and is explained as follows. As explained above, the hardware of Dorst outputs device control codes dependent upon the memory which is to be read or written (which is indicated in some form by the request codes). When the hardware of Dorst is replaced by the microcode of FOLDOC, the combination would entail microcode outputting control codes dependent upon these request codes. Therefore, the execution of the

microcode (which necessarily involves executing a first instruction of the microcode, which is at a start address) is dependent upon receiving a request code. Moreover, the execution of a specific microcode (which must specifically reference the particular register set with timing parameters that correlates to the memory to be accessed) is dependent upon a particular request code (which denotes the memory to be accessed). Finally, it is inherent that microcode is stored in some form of microcode memory in order to be run. Therefore, the combination teaches the overall limitations of executing a microcode program stored in a microcode memory, wherein the execution of the microcode program being dependent upon the first request code in that the start address of the microcode program corresponds at least partly to the first request code.

18. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over Cataldo, Dorst, FOLDOC, and Tanenbaum as applied to claim 17 above, and further in view of Berggreen (US 20030016685 A1).

19. Consider claim 18, neither Cataldo nor Dorst nor FOLDOC nor Tanenbaum disclose a quantity of the plurality of access points from which the request is received by the interface engine from the pipeline is adjustable based on one of i) a data flow rate through the pipeline, and ii) a flow capacity of the external device.

On the other hand, Berggreen does disclose of the concept of processing packets in different manners depending on one of i) a data flow rate through the pipeline and ii) a flow capacity of the external device ([0038]-[0039], for example, a first

operating mode is used when the data flow rate is low and a second operating mode is used when the data flow rate is high).

Berggreen's teaching increases flexibility by catering to a wide range of configurations and environments in part on current and historical operating conditions while reducing congestion and maintaining QoS for all message types (Berggreen, [0006]-[0007]).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teaching of Berggreen with invention of Cataldo, Dorst, FOLDLOC, and Tanenbaum in order to increase flexibility by catering to a wide range of configurations and environments in part on current and historical operating conditions while reducing congestion and maintaining QoS for all message types. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that Berggreen's teaching of processing packets in differing ways dependent on data flow rate, when applied to the invention of Cataldo and Dorst, encompasses the possibility that a memory access needed for one mode is not needed for another. Additionally, the second mode of Berggreen involves storing received messages in a second queue which does not happen in the first mode, which is one example of how rate would determine whether a memory access occurs from a specific access point of the programmable pipeline or not. It would have been readily recognized to one of ordinary skill in the art at the time of the invention that the environment of Berggreen regarding QoS is analogous to the invention of Cataldo which can be programmed to do traffic shaping.

***Response to Arguments***

20. Applicant argues on page 14-15 that Dorst fails to teach or suggest a request comprising a first request code, and states that the first request code of the present invention relates to the pipeline from which the request is sent and not to an external device to be accessed in Dorst. However, the request code in Dorst can be read to relate to the pipeline as the request code is not concerned with the specific memory access signals which are provided by the memory controller, such as the read enable signal and so forth. Moreover, although this specific language is not claimed, it is noted that "related" can be read broadly; in other words, the request code of the present invention does relate to an external device to be accessed in that the request code does cause the external device to be accessed.

21. Applicant argues from pages 15-20 that Dorst does not teach of a microcode program. In response, examiner has introduced an art which explicitly discloses of microcode, and an art which explicitly conveys the examiner's belief that the use of software instead of hardware to implement an aspect of the prior art combination would not be a novel difference.

***Conclusion***

22. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Keith Vicary whose telephone number is (571)270-1314.

The examiner can normally be reached on Monday - Thursday, 6:15 a.m. - 5:45 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on 571-272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Keith Vicary/  
Examiner, Art Unit 2183

/Eddie P Chan/  
Supervisory Patent Examiner, Art Unit 2183